CS 603: Programming Language Organization

Lecture 13

Spring 2005

Department of Computer Science

University of Alabama

Joel Jones

Outline

- Questions
- μ-Scheme (cont.)
- Reading for next time

Simple higher-order functions

- Composition
  - (define o (f g) (lambda (x) (f (g x))))
  - (define even? (n) (= 0 (mod n 2)))
  - (val odd? (o not even?))

Pair Up:

• Write a function (to8th) using composition with square and ? that raises the input to the 8th power

```
-> (define square(x) (* x x))
square
-> (val to8th …)
<procedure>
-> (to8th 2)
256
```

Higher-order functions on lists

- Filter –

```
(define filter (p? l)
    (if (null? l) '()
        (if (p? (car l))
            (cons (car l) (filter p? (cdr l)))
            (filter p? (cdr l)))))
```

- Exists? –

```
(define exists? (p? l)
    (if (null? l) #f
        (if (p? (car l))
            #t
            (exists? p? (cdr l)))))
```

Higher-order functions on lists (cont.)

- All? –

```
(define all? (p? l)
    (if (null? l) #t
        (if (p? (car l))
            (exists? p? (cdr l))
            #f)))
```

- Map –

```
(define map (f l)
    (if (null? l) '()
        (cons (f (car l)) (map f (cdr l)))))
```

Pair Up:

- How are the arguments of `filter`, `exists?`, and `all?` different from those of `map?`

Higher-order functions on lists (cont.)

- Foldr
- Foldl